


[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☒ The ACM Digital Library ☐ The Guide

table searches updates inserts performed concurrently altern

SEARCH


[Feedback](#) [Report a problem](#) [Satisfaction survey](#)

Terms used

table searches updates inserts performed concurrently alternating slots timing signal

 Found 50,407 of
155,867

 Sort results
by

 relevance ☒

[Save results to a Binder](#)

 Try an [Advanced Search](#)

 Display
results

 expanded form ☒

[Search Tips](#)

 Try this search in [The ACM Guide](#)
☐ Open results in a new
window

Results 1 - 20 of 200

 Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

 Relevance scale ☐ ☐ ☐ ☐ ☐

1 [Special issue: AI in engineering](#)

D. Sriram, R. Joobhani

 January 1985 **ACM SIGART Bulletin**, Issue 91

 Full text available: [pdf\(8.79 MB\)](#)

 Additional Information: [full citation](#), [abstract](#)

The papers in this special issue were compiled from responses to the announcement in the July 1984 issue of the SIGART newsletter and notices posted over the ARPAnet. The interest being shown in this area is reflected in the sixty papers received from over six countries. About half the papers were received over the computer network.

2 [Fast detection of communication patterns in distributed executions](#)

Thomas Kunz, Michiel F. H. Seuren

 November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research**

 Full text available: [pdf\(4.21 MB\)](#)

 Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

3 [Human-computer interface development: concepts and systems for its management](#)

H. Rex Hartson, Deborah Hix

 March 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 1

 Full text available: [pdf\(7.97 MB\)](#)

 Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

Human-computer interface management, from a computer science viewpoint, focuses on the process of developing quality human-computer interfaces, including their representation, design, implementation, execution, evaluation, and maintenance. This survey presents important concepts of interface management: dialogue independence, structural modeling, representation, interactive tools, rapid prototyping, development methodologies, and control structures. *Dialogue independence* is th ...

4 [Distributed operating systems](#)

Andrew S. Tanenbaum, Robbert Van Renesse
December 1985 **ACM Computing Surveys (CSUR)**, Volume 17 Issue 4


Full text available:  pdf(5.49 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

Distributed operating systems have many aspects in common with centralized ones, but they also differ in certain ways. This paper is intended as an introduction to distributed operating systems, and especially to current university research about them. After a discussion of what constitutes a distributed operating system and how it is distinguished from a computer network, various key design issues are discussed. Then several examples of current research projects are examined in some detail ...

5 Pen computing: a technology overview and a vision

André Meyer
July 1995 **ACM SIGCHI Bulletin**, Volume 27 Issue 3


Full text available:  pdf(5.14 MB)

Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

This work gives an overview of a new technology that is attracting growing interest in public as well as in the computer industry itself. The visible difference from other technologies is in the use of a pen or pencil as the primary means of interaction between a user and a machine, picking up the familiar pen and paper interface metaphor. From this follows a set of consequences that will be analyzed and put into context with other emerging technologies and visions. Starting with a short historic ...

6 Software pipelining

Vicki H. Allan, Reese B. Jones, Randall M. Lee, Stephen J. Allan
September 1995 **ACM Computing Surveys (CSUR)**, Volume 27 Issue 3


Full text available:  pdf(4.72 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Utilizing parallelism at the instruction level is an important way to improve performance. Because the time spent in loop execution dominates total execution time, a large body of optimizations focuses on decreasing the time to execute each iteration. Software pipelining is a technique that reforms the loop so that a faster execution rate is realized. Iterations are executed in overlapped fashion to increase parallelism. Let $\{ABC\}^n$
Keywords: instruction level parallelism, loop reconstruction, optimization, software pipelining

7 The NYU Ultracomputer—designing a MIMD, shared-memory parallel machine (Extended Abstract)

Allan Gottlieb, Ralph Grishman, Clyde P. Kruskal, Kevin P. McAuliffe, Larry Rudolph, Marc Snir
April 1982 **Proceedings of the 9th annual symposium on Computer Architecture**

Full text available:  pdf(1.36 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

We present the design for the NYU Ultracomputer, a shared-memory MIMD parallel machine composed of thousands of autonomous processing elements. This machine uses an enhanced message switching network with the geometry of an Omega-network to approximate the ideal behavior of Schwartz's paracomputer model of computation and to implement efficiently the important fetch-and-add synchronization primitive. We outline the hardware that would be required to build a 4096 processor system using 1990' ...

8 Iterative schedule optimization for voltage scalable distributed embedded systems

Marcus T. Schmitz, Bashir M. Al-Hashimi, Petru Eles
February 2004 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 3 Issue

1

Full text available:  pdf(273.28 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

We present an iterative schedule optimization for multirate system specifications, mapped onto heterogeneous distributed architectures containing dynamic voltage scalable processing elements (DVS-PEs). To achieve a high degree of energy reduction, we formulate a generalized DVS problem, taking into account the power variations among the executing tasks. An efficient heuristic is presented that identifies optimized supply voltages by not only "simply" exploiting slack time, but under the addition ...

Keywords: Dynamic voltage scaling, embedded systems, energy minimization, heterogeneous distributed systems, scheduling, system synthesis

9 The NYU ultracomputer—designing a MIMD, shared-memory parallel machine

Allan Gottlieb, Ralph Grishman, Clyde P. Kruskal, Kevin P. McAuliffe, Larry Rudolph, Marc Snir
August 1998 **25 years of the international symposia on Computer architecture (selected papers)**

Full text available:  pdf(1.74 MB) Additional Information: [full citation](#), [references](#), [index terms](#)

10 A high performance transparent bridge

Martina Zitterbart, Ahmed N. Tantawy, Dimitrios N. Serpanos
August 1994 **IEEE/ACM Transactions on Networking (TON)**, Volume 2 Issue 4

Full text available:  pdf(1.41 MB) Additional Information: [full citation](#), [references](#), [index terms](#)

11 Query evaluation techniques for large databases

Goetz Graefe
June 1993 **ACM Computing Surveys (CSUR)**, Volume 25 Issue 2

Full text available:  pdf(9.37 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

Database management systems will continue to manage large data volumes. Thus, efficient algorithms for accessing and manipulating large sets and sequences will be required to provide acceptable performance. The advent of object-oriented and extensible database systems will not solve this problem. On the contrary, modern data models exacerbate the problem: In order to manipulate large sets of complex objects as efficiently as today's database systems manipulate simple records, query-processi ...

Keywords: complex query evaluation plans, dynamic query evaluation plans, extensible database systems, iterators, object-oriented database systems, operator model of parallelization, parallel algorithms, relational database systems, set-matching algorithms, sort-hash duality

12 Cycles to recycle: garbage collection to the IA-64

Richard L. Hudson, J. Elliot Moss, Sreenivas Subramoney, Weldon Washburn
October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management**, Volume 36 Issue 1


Full text available:  pdf(1.25 MB) Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

The IA-64, Intel's 64-bit instruction set architecture, exhibits a number of interesting architectural features. Here we consider those features as they relate to supporting garbage collection (GC). We aim to assist GC and compiler implementors by describing how one may

exploit features of the IA-64. Along the way, we record some previously unpublished object scanning techniques, and offer novel ones for object allocation (suggesting some simple operating system support that would simplify it ...

13 Parallel execution of prolog programs: a survey

Gopal Gupta, Enrico Pontelli, Khayri A.M. Ali, Mats Carlsson, Manuel V. Hermenegildo
July 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**,
Volume 23 Issue 4

Full text available:  pdf(1.95 MB)


Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Since the early days of logic programming, researchers in the field realized the potential for exploitation of parallelism present in the execution of logic programs. Their high-level nature, the presence of nondeterminism, and their referential transparency, among other characteristics, make logic programs interesting candidates for obtaining speedups through parallel execution. At the same time, the fact that the typical applications of logic programming frequently involve irregular computatio ...

Keywords: Automatic parallelization, constraint programming, logic programming, parallelism, prolog

14 Technical correspondence


CORPORATE Tech Correspondence
October 1989 **Communications of the ACM**, Volume 32 Issue 10

Full text available:  pdf(2.15 MB)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

15 Atomic heap transactions and fine-grain interrupts

Olin Shivers, James W. Clark, Roland McGrath
September 1999 **ACM SIGPLAN Notices , Proceedings of the fourth ACM SIGPLAN international conference on Functional programming**, Volume 34 Issue 9

Full text available:  pdf(1.45 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Languages such as Java, ML, Scheme, and Haskell provide automatic storage management, that is, garbage collection. The two fundamental operations performed on a garbage-collected heap are "allocate" and "collect." Because the heap is in an inconsistent state during these operations, they must be performed atomically. Otherwise, a heap client might access the heap during a time when its fundamental invariants do not hold, corrupting the heap. Standard techniques for providing this atomicity guaran ...

16 Status report of the graphic standards planning committee

Computer Graphics staff
August 1979 **ACM SIGGRAPH Computer Graphics**, Volume 13 Issue 3

Full text available:  pdf(15.01 MB)

Additional Information: [full citation](#), [references](#), [citations](#)

17 The white dwarf: a high-performance application-specific processor

A. Wolfe, M. Breternitz, C. Stephens, A. L. Ting, D. B. Kirk, R. P. Bianchini, J. P. Shen
May 1988 **ACM SIGARCH Computer Architecture News , Proceedings of the 15th Annual International Symposium on Computer architecture**, Volume 16 Issue 2

Full text available:  pdf(1.40 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper presents the design and implementation of a high-performance special-purpose processor, called The White Dwarf, for accelerating finite element analysis algorithms. The White Dwarf CPU contains two Am29325 32-bit floating-point processors and one Am29332 32-bit ALU, and employs a wide-instruction word architecture in which the application algorithm is directly implemented in microcode. The entire system is VME-bus compatible and interfaces with a SUN 31160 host. The syste ...

18 Prefetching using Markov predictors

Doug Joseph, Dirk Grunwald

May 1997 **ACM SIGARCH Computer Architecture News , Proceedings of the 24th annual international symposium on Computer architecture**, Volume 25 Issue 2

Full text available:  [pdf\(1.68 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Prefetching is one approach to reducing the latency of memory operations in modern computer systems. In this paper, we describe the *Markov prefetcher*. This prefetcher acts as an interface between the on-chip and off-chip cache, and can be added to existing computer designs. The Markov prefetcher is distinguished by prefetching *multiple reference predictions* from the memory subsystem, and then prioritizing the delivery of those references to the processor. This design results in a pr ...

19 Novel ideas: A design space evaluation of grid processor architectures

Ramadass Nagarajan, Karthikeyan Sankaralingam, Doug Burger, Stephen W. Keckler

December 2001 **Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture**



Full text available:  [pdf\(1.29 MB\)](#)  Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)
[Publisher Site](#)

In this paper, we survey the design space of a new class of architectures called Grid Processor Architectures (GPAs). These architectures are designed to scale with technology, allowing faster clock rates than conventional architectures while providing superior instruction-level parallelism on traditional workloads and high performance across a range of application classes. A GPA consists of an array of ALUs, each with limited control, connected by a thin operand network. Programs are executed b ...

20 The multicluster architecture: reducing cycle time through partitioning

Keith I. Farkas, Paul Chow, Norman P. Jouppi, Zvonko Vranesic

December 1997 **Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture**

Full text available:  [pdf\(1.44 MB\)](#)  Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)
[Publisher Site](#)

The multicluster architecture that we introduce offers a decentralized, dynamically-scheduled architecture, in which the register files, dispatch queue, and functional units of the architecture are distributed across multiple clusters, and each cluster is assigned a subset of the architectural registers. The motivation for the multicluster architecture is to reduce the clock cycle time, relative to a single-cluster architecture with the same number of hardware resources, by reducing the size and ...

Keywords: decentralized architecture, partitioned architecture, static instruction scheduling, register allocation

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)